6/5/1
DIALOG(R)File 351:Derwent WPI

011718460
WPI Acc No: 1998-135370/ **199813**
XRPX Acc No: N98-107194

Monitoring method for system having multi-processors - by assigning predictor of capacity of system that responds quickly as evaluation object to enable allocation of processes to several processors of system whose loaded condition is monitored

Patent Assignee: MATSUSHITA ELECTRIC WORKS LTD (MATW)
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
JP 10011407	A	19980116	JP 96164937	A	19960625	199813	B

Priority Applications (No Type Date): JP 96164937 A 19960625
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 10011407	A		5	G06F-015/16	

Abstract (Basic): JP 10011407 A

The method involves monitoring the loaded condition of a system, that has several processors and does distributed processing, based on the priority of each process. The allocation of the processes to several processors is done by assigning the predictor of the capacity of the system that responds quickly as an evaluation object.

ADVANTAGE - Allows efficient allocation of processes to processors. Describes true behaviour of each CPU correctly. Improves overall performance and overall throughput of system.

Dwg.0/5

Title Terms: MONITOR; METHOD; SYSTEM; MULTI; PROCESSOR; ASSIGN; PREDICT; CAPACITY; SYSTEM; RESPOND; QUICK; EVALUATE; OBJECT; ENABLE; ALLOCATE; PROCESS; PROCESSOR; SYSTEM; LOAD; CONDITION; MONITOR
Index Terms/Additional Words: CENTRAL; PROCESSING; UNIT
Derwent Class: T01
International Patent Class (Main): G06F-015/16
File Segment: EPI

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－11407

(43)公開日　平成10年(1998)１月16日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　15/16	３８０		Ｇ０６Ｆ　15/16	３８０Ｚ	
	３７０			３７０Ｎ	
	４６０			４６０Ｚ	

審査請求　未請求　請求項の数４　ＯＬ　（全５頁）

(21)出願番号　特願平8－164937

(22)出願日　平成８年(1996)６月25日

(71)出願人　000005832
松下電工株式会社
大阪府門真市大字門真1048番地

(72)発明者　マイケル　ミグドル
大阪府門真市大字門真1048番地　松下電工株式会社内

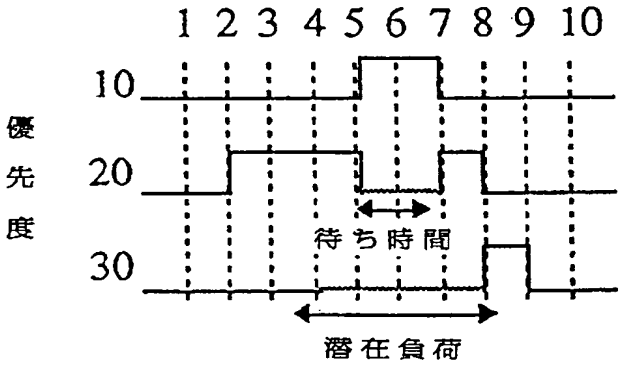
(74)代理人　弁理士　倉田　政彦

(54)【発明の名称】　マルチプロセッサの監視方法

(57)【要約】

【課題】複数のプロセッサを用いて分散処理を行うシステムの負荷状態を各処理の優先度に基づいて監視し、複数のプロセッサの間で効率的に処理を分配するためのマルチプロセッサの監視方法を提供する。

【解決手段】複数のプロセッサを用いて分散処理を行うシステムの負荷状態を各処理の優先度に基づいて監視する方法において、事象に対して速やかに応答するシステムの能力の予測値、又は、プロセッサを必要とするタスク群に十分なプロセッサの時間を割り当てるシステムの能力の予測値、又は、より優先度の高いタスクによる現時点でのシステムの利用状況のうち少なくとも１つ以上に基づいて複数のプロセッサの間で効率的に処理を分配する。



1

【特許請求の範囲】

【請求項１】　複数のプロセッサを用いて分散処理を行うシステムの負荷状態を各処理の優先度に基づいて監視する方法において、事象に対して速やかに応答するシステムの能力の予測値を評価対象とすることを特徴とするマルチプロセッサの監視方法。

【請求項２】　複数のプロセッサを用いて分散処理を行うシステムの負荷状態を各処理の優先度に基づいて監視する方法において、プロセッサを必要とするタスク群に、十分なプロセッサの時間を割り当てるシステムの能力の予測値を評価対象とすることを特徴とするマルチプロセッサの監視方法。

【請求項３】　複数のプロセッサを用いて分散処理を行うシステムの負荷状態を各処理の優先度に基づいて監視する方法において、より優先度の高いタスクによる現時点でのシステムの利用状況を評価対象とすることを特徴とするマルチプロセッサの監視方法。

【請求項４】　請求項１乃至３の評価対象のうち少なくとも１つ以上に基づいて複数のプロセッサの間で効率的に処理を分配することを特徴とするマルチプロセッサの監視方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、複数のプロセッサを用いて分散処理を行うシステムの負荷状態を各処理の優先度に基づいて監視し、複数のプロセッサの間で効率的に処理を分配するためのマルチプロセッサの監視方法に関するものである。

【０００２】

【従来技術】従来、優先度に基づいてオペレーティング・システムの負荷状態をモニターするための現在の方法は、一般的に生のＣＰＵの利用率（％）又は待機しているジョブの数を用いている。ジョブの分配法を選択する方法としては、最も負荷の軽いＣＰＵにジョブを送る方法や、各ＣＰＵに順番にジョブを送る方法、無負荷のＣＰＵにジョブを要求する一斉同報をさせる方法を含めて、多くの方法が存在する。

【０００３】これらのすべての方法の問題は、ＣＰＵの負荷の数値に依存することである。実行中のタスクの優先度が無視されているので、例えば、高優先度のタスクを実行するのに、自己の時間の８０％を費やしているシステムと、低優先度のタスクを実行するのに自己の時間の８０％を費やしているシステムとの間に差が生じない。その差というのは、第１の場合では、中優先度のタスクにとってはＣＰＵの負荷が重いと見えるのに対し、第２の場合では、中優先度のタスクにとってはＣＰＵの負荷が軽いと見えることである。このことは、与えられたジョブをどちらのＣＰＵに実行させるべきかを選択するときに、不適切な選択を招き、その結果、全体のシステムの効率が悪くなる。

2

【０００４】そこで、タスクの優先度に基づいて複数のプロセッサを監視する方法として、例えば、特開平７－２８２０１３号公報に開示された方法がある。

【０００５】

【発明が解決しようとする課題】本発明の目的は、複数のプロセッサの負荷をタスクの優先度を考慮して監視する方法において、上述の従来技術には無かった３種類の評価方法を提案し、これらの方法に基づいて効率的にジョブの分配を可能とすることにある。

【０００６】

【課題を解決するための手段】本発明のシステムでは、実行中のタスクの優先度を考慮に入れたＣＰＵの負荷状態の新規な計測手段を開発することによって、従来例の問題点を克服している。この計測手段は、潜在負荷（事象に対して速やかに応答するシステムの能力の予測値）、タスク負荷（ＣＰＵを必要とするタスク群に、十分なＣＰＵの時間を割り当てるシステムの能力の予測値）、又は、利用率（より優先度の高いタスクによる現時点でのシステムの利用状況）という用語で表現され得る。各ＣＰＵは、カーネルのレベルでのタスクのステータスの変化をモニターすることによって、自分自身の計測を続ける。その計測の結果は、すべてのＣＰＵに対する、より効率的なジョブの分配を選択するために利用され得る。

【０００７】以下、本発明のシステムの概略を説明する。システムの心臓部は、オペレーティング・システムのカーネルに付加された１つのソフトウェアであり、これは、タスクのステータスの変化をモニターする。次に、この情報は、３つの異なるアルゴリズムのうちの１つに伝達される。これらのアルゴリズムは、各優先レベルにおけるタスクから見た利用状況を計測し、潜在負荷を予測し、タスク負荷を予測する。各システムの個々の要求に応じて、ＣＰＵの負荷を評価するために、これらの計測手段のうちの任意の組み合わせを利用することができる。各行が優先度の数値とその優先度に関連した負荷の数値を含むデータで構成された「負荷テーブル」が出力される。

【０００８】次に、ＣＰＵの情報テーブルは、ＣＰＵが密結合（例えば、同じＰＣ基板上でＩＳＡバスを介して通信する複数のＣＰＵ）であるか、疎結合（例えば、Ｅｔｈｅｒｎｅｔを介して通信する複数のＣＰＵ）であるかに応じて、２通りの方法のうちいずれかで処理される。密結合の場合、各ＣＰＵは一斉同報送信を用いた更新により自己のステータスを他のノードに知らせる。そして、各ＣＰＵは自己のタスクのうちの１つ又はそれ以上を異なるＣＰＵに割り当てるか否かを選択することについて責任を持つ。疎結合の場合には、１つのＣＰＵがすべての情報の管理に責任を持ち、他のＣＰＵに対してジョブの分配方法を告げるマスターとして動作する。

【０００９】

【発明の実施の形態】システムの動作を簡単な例により説明する。まず、第１に、１つのＣＰＵについての負荷テーブルの計算について示す。図１は３つのタスクが実行されている簡単なケースを示す。このシステムでは、低い値の優先度は優先度の高いタスクを示している。言い換えれば、優先度１０のタスクは優先度２０のタスクよりも常に先にＣＰＵを受け取る。最初に、各タスクは、休止状態であって、ＣＰＵを必要としない。時間２において、優先度２０のタスクが実行可能になると、ＣＰＵを直ちに受け取る。このタスクは４ユニットの実行時間を必要とする。時間４において、優先度３０のタスクが実行可能になるが、優先度のより高いタスクがＣＰＵを使用しているので、このタスクは待機状態に入る。時間５において、優先度１０のタスクが実行可能状態となる。このタスクは、優先度２０のタスクよりも優先度

が高いので、直ちにＣＰＵを受け取る。優先度２０のタスクは、先に実行権を得ており、まだ１ユニットの実行時間を残しているので、待機状態に入る。時間７において、優先度１０のタスクが終了すると、優先度２０のタスクは実行を再開し、時間８において終了する。この時点で、優先度３０のタスクは、最後にＣＰＵを受け取り、１ユニットの時間後に、その仕事を終了する。

【００１０】まず、タイミングデータは、表１に示されるような単純なテーブルに変換される。このテーブルは、３つの負荷計測／性能予測のセクションの各々で使用される。テーブルの行方向は優先度、列方向は時間を表わし、は実行中、×は非実行、Ｗは待機中を意味する。

【００１１】

【表１】

	1	2	3	4	5	6	7	8	9	10
10	X	X	X	X	X	O	O	X	X	X
20	X	X	O	O	O	W	W	O	X	X
30	X	X	X	X	W	W	W	W	O	X

【００１２】　利用率計測のセクション

利用率計測の計算をするために、優先レベルの各々に対するＣＰＵの利用状況が調べられる。この例では、ＣＰＵは時間１、２及び１０の間を除いて、すなわち、７０％の時間は使用されていた。しかしながら、優先度３０のタスクにとっては、ＣＰＵは６０％の時間のみ利用不可能であった。同様に、優先度２０のタスクにとっては、２０％の時間だけＣＰＵは使用され、８０％の時間は利用可能であった。したがって、ＣＰＵの利用率計測モジュールの出力は、表２に示されるようなテーブルとなるであろう。優先度の値が２５５に対応する負荷は、生のＣＰＵの利用率に対応していることに注意されたい。

【００１３】

【表２】

優先度	ＣＰＵ利用状況
１０	０％
２０	２０％
３０	６０％
２５５	７０％

【００１４】　潜在負荷の予測計算のセクション

潜在負荷を予測計算するために、或る事象が異なる時点の各々で起こったとしたときに観測されるであろう潜在負荷が計算される。例えば、優先度３０のタスクが時間１又は２において実行を要求されると、このタスクは直ちにＣＰＵを受け取るであろう。反対に、このタスクが時間５においてＣＰＵを必要とした場合には、３ユニットの時間は待たなくてはならなかったであろう。それ故、１０ユニットの時間にわたる平均的な潜在負荷は、（０＋０＋６＋５＋４＋３＋２＋１＋０＋０）／１０＝２．１ユニットの時間となる。優先度２０のタスクに対する平均的な潜在負荷は、（０＋０＋０＋０＋２＋１＋０＋０＋０）／１０＝０．３ユニットの時間となる。優先度１０のタスクに対する平均的な潜在負荷は０である。なぜなら、ＣＰＵを使用していた、それよりも上位のタスクは存在しないから。潜在負荷の予測モジュールの出力は、表３に示されるようになるであろう。

【００１５】

【表３】

優先度	予測潜在負荷
１０	０
２０	０．３
３０	２．１
２５５	３．２

【００１６】　待ち時間の予測計算セクション

待ち時間の予測計算のために、或る事象が異なる時点の各々で起こったとしたときに観測されるであろう待ち時間が計算される。例えば、優先度２０のタスクが１、２又は７の時間でスタートすれば、待ち時間は０であろう。他の場合には、２ユニットの待ち時間となるであろう。故に、待ち時間の予測値は、（０＋０＋２＋２＋２＋２＋０）／７＝１．１４ユニットの時間となる。時間８以降でのスタートについては、計算できないことに注意されたい。なぜなら、タスクが現在の時間とオーバーラップするであろうから。また、潜在時間は待ち時間の計算には含まれていないことに注意されたい。最後に、最も低い優先度の計算については、使用されたタスクの期間は、優先度３０のタスクの時間（１ユニット）と同じであると仮定され、それ故、待ち時間は依然として０であることに注意されたい。この例の出力テーブルは、

表４のようになるであろう。

【００１７】

【表４】

優先度	予測待ち時間
１０	０
２０	１．４１
３０	０
２５５	０

【００１８】各ＣＰＵは、上に説明した情報を管理しており、周期的にそれを更新したり、古い情報を廃棄したりする。これらの数値は、２つの主な目的を有する。１つは単にＣＰＵ負荷状態のモニターとしてであり、それは、システムのパフォーマンスの異常を検出するのに有益であり得る。他の目的は、負荷均衡化システムの入力としてである。

【００１９】このシステムには、２つの基本的な形態がある。１つは、ＣＰＵ間の通信が低速な状況において有益であるが、そのスケジューリングの方針については正確さにおいて劣る。第２のものは、高速なプロセッサ間通信が可能なときに有益であり、プロセッサ間で交換される情報量が増加することにより、より正確なものとなる。以下、それぞれの場合の実施例について説明する。

【００２０】

【実施例】

（Ａ）低速法の実施例

この方法では、その全体的な負荷が所定のスレショルドよりも少ないプロセッサ群のみが、それらの負荷テーブル（表２に示されるような）を他のすべてのノードに一斉同報送信する。そのとき、負荷の重いノードは、自己が知る無負荷のノードについての情報に従って、自己のタスク群の１つを引き継ぐことをそのノードに要求する。

【００２１】一例として、図２に示される簡単な例を考える。この例は、３つのノードを有しており、負荷評価システムは、ＣＰＵ利用率のみを用いる。この例では、ノード１は、その上で実行される３つのタスクを有している。高優先度のタスクはＣＰＵの時間の２０％を取っており、中優先度のタスクはＣＰＵの時間の１０％を取っており、低優先度のタスクはＣＰＵの時間の６０％を取っている。ノード２及び３の負荷は軽く、それ故、これら両者は、現時点の自己の利用率テーブルを一斉同報送信する。ノード１は中優先度のジョブを他のノードに送りたい。なぜなら、その高優先度のジョブが中優先度のジョブについて緩慢な応答を引き起こしているからである。従来の負荷均衡化システムでは、全体の利用率の値が小さい方のノード２にジョブを送る傾向があった。このことは、実際には、中優先度のノードのパフォーマンスの低下を招いている。なぜなら、ノード２の高優先度のプロセスはＣＰＵの時間の３０％を取っているから

である。しかし、中優先度のジョブは、ノード３上では、もっと良い応答を得られるであろうということをノード１は理解している。そして、それ故に、図３に示されるように、ジョブをノード３に送る。現実の利用分布は多く変更されていないけれども、中優先度のタスクのパフォーマンスは、ノード３にシフトすることによって、かなりの程度、改善されるに違いないことに注意されたい。

【００２２】（Ｂ）高速法の実施例

この方法では、各プロセッサは、自己のタイミング情報を（負荷測定情報の代わりに）マスタープロセッサに送る。そのタイミング情報は、（表１に示されるような）タイミングテーブルを含み、また、将来のプロセッサのタスク群についてのタイミング要求について既知の任意の情報をも含んでいる。このマスタープロセッサは、送られてきたタイミングプロフィールを検討して、自己が受信した情報に基づいて、すべてのプロセッサのために最適のジョブの配置を決定する。タイミング情報を送信するのに必要な帯域幅のために、高速のプロセス間通信が必要である。この結果、より効率的な配置を選択することができ、全体的なパフォーマンスを改善できる。

【００２３】この方法は、図４に示される。ノード１はマスターのノードであり、自分自身のタイミングデータを処理すると共に、ノード２及び３から送られてきたタイミングデータも処理する。マスターが一度データを処理すると、図５に示されるように、マスターは選択された配置を一斉同報送信し、そのとき、各ノードは、マスターのノードによって選択されたように、ジョブを引き継ぐ。

【００２４】

【発明の効果】このシステムでは、負荷計算において実行中のタスクの優先度を用いることによって、生のＣＰＵの利用と優先度に基づくＣＰＵの利用との間に明瞭な区別をつけることができ、より正確に各ＣＰＵの真の振る舞いを記述できる。その結果、複数のタスクは、それらの優先度に基づいて、より正確に分配され、システムのパフォーマンスと全体的なスループットを改善できる。

【図面の簡単な説明】

【図１】本発明の監視方法によるマルチタスクの監視状況を示す説明図である。

【図２】本発明の監視方法を低速なプロセッサ通信によるマルチプロセッサシステムに適用した場合のジョブの再分配前の状態を示す説明図である。

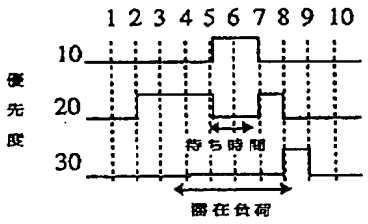
【図３】本発明の監視方法による低速なプロセッサ通信によるマルチプロセッサシステムに適用した場合のジョブの再分配後の状態を示す説明図である。

【図４】本発明の監視方法を高速なプロセッサ通信によるマルチプロセッサシステムに適用した場合のマスタープロセッサへの情報の流れを示す説明図である。

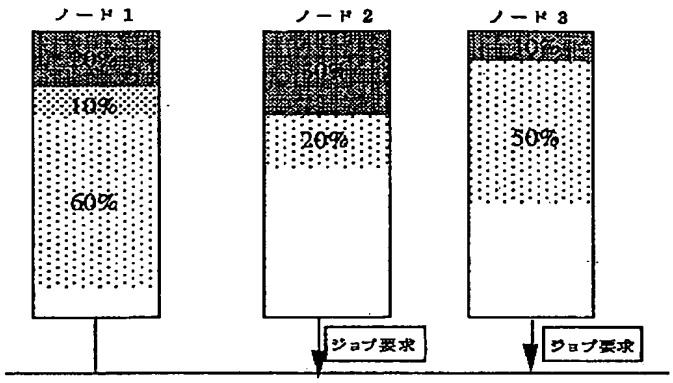
【図５】本発明の監視方法を高速なプロセッサ通信によるマルチプロセッサシステムに適用した場合のマスタープロセッサからの情報の流れを示す説明図である。

【符号の説明】
１～３　ノード

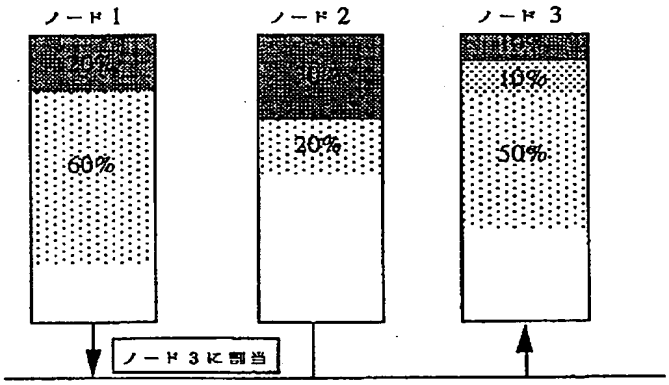
【図１】



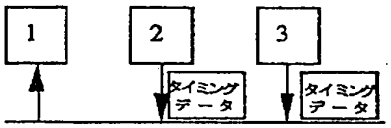
【図２】



【図３】



【図４】



【図５】